Chapter 2
Finite Automata

2.1 Introduction

- Finite automata: a first model of the notion of effective procedure. (They also have many other applications).

- The concept of finite automaton can be derived by examining what happens when a program is executed on a computer: state, initial state, transition function.

- The finite state hypothesis and its consequences: finite or cyclic sequences of states.

- The problem of representing the data: only a finite number of different data sets can be represented since there exists only a finite number of initial states.

Representing data.

- Problem: to recognize a language.
- Data: a word.
- We will assume that the word is fed to the machine character by character, one character being handled at each cycle and the machine stopping once the last character has been read.

2.2 Description

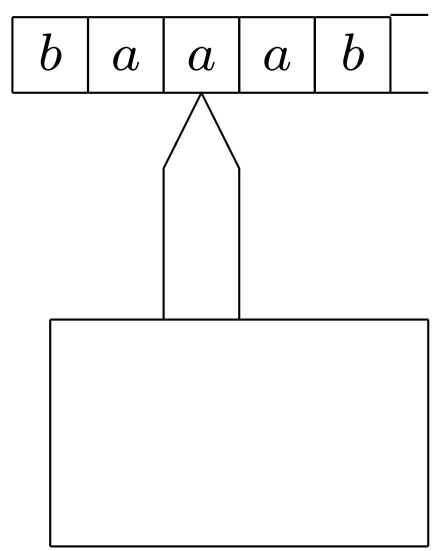
- Input tape.

- Set of states:
  - initial state,
  - accepting states.

- Execution step:

tape :

head :



2.3 Formalization

A deterministic finite automaton is defined by a five-tuple $M = (Q, \Sigma, \delta, s, F)$, where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $s \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states.

Defining the language accepted by a finite automaton

- Configuration : $(q, w) \in Q \times \Sigma^*$.
- Configuration derivable in one step: $(q, w) \vdash_M (q', w')$.
- Derivable configuration (multiple steps) : $(q, w) \vdash_M^* (q', w')$.
- Execution:

$$(s, w) \vdash (q_1, w_1) \vdash (q_2, w_2) \vdash \cdots \vdash (q_n, \varepsilon)$$

- Accepted word:

$$(s, w) \vdash_M^* (q, \varepsilon)$$

  and $q \in F$.

- Accepted language $L(M)$:

$$\{w \in \Sigma^* \mid (s, w) \vdash_M^* (q, \varepsilon) \text{ avec } q \in F\}.$$

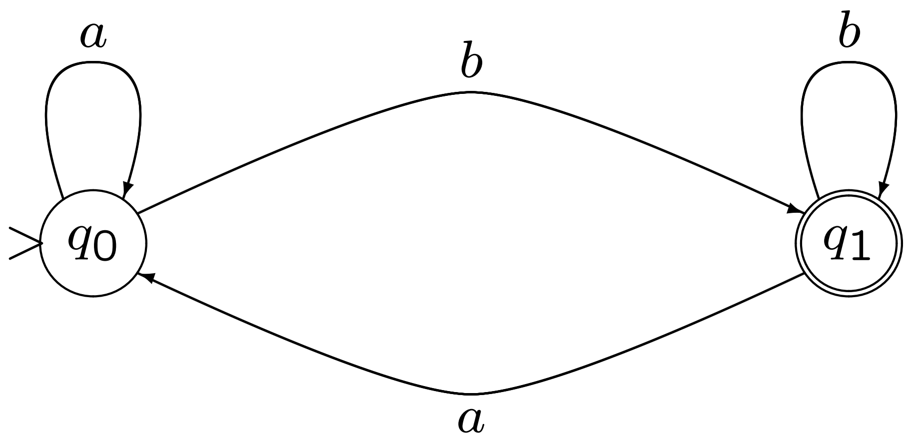
2.4 Examples

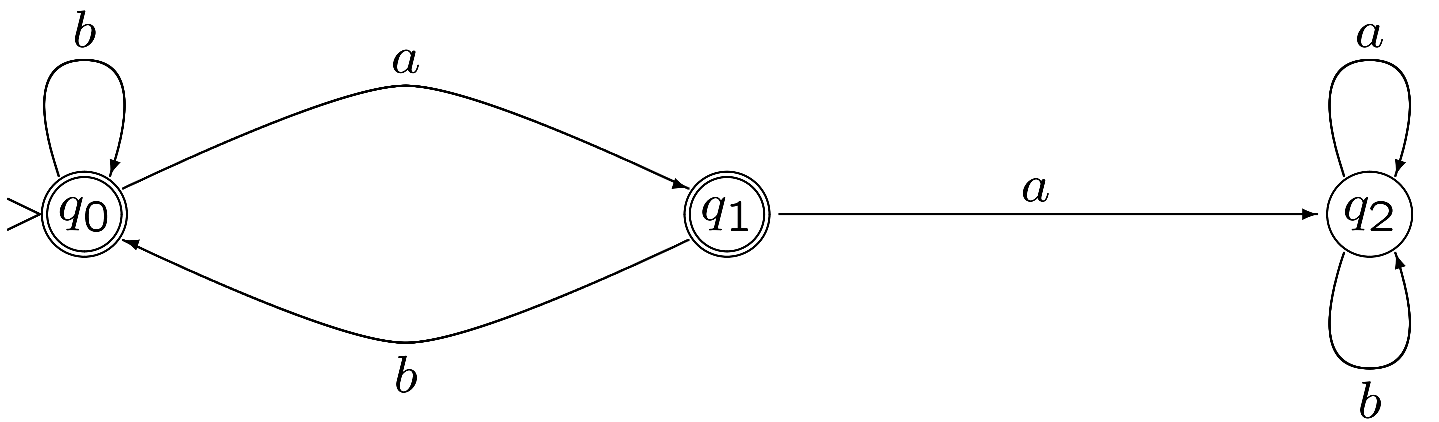
Words ending with b :

δ :

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_1

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_1\}$





$\{w \mid w$ does not contain 2 consecutive a's$\}$.

2.5 Nondeterministic finite automata

Automata that can *choose* among several transitions.

Motivation :

- To examine the consequences of generalizing a given definition.
- To make describing languages by finite automata easier.
- The concept of non determinism is generally useful.

Description

Nondeterministic finite automata are finite automata that allow:

- several transitions for the same letter in each state,
- transitions on the empty word (i.e., transitions for which nothing is read),
- transitions on words of length greater than 1 (combining transitions).

Nondeterministic finite automata accept if a least one execution accepts.

Formalization

A nondeterministic finite automaton is a five-tuple $M = (Q, \Sigma, \Delta, s, F)$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\Delta \subset (Q \times \Sigma^* \times Q)$ is the transition relation,
- $s \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states.

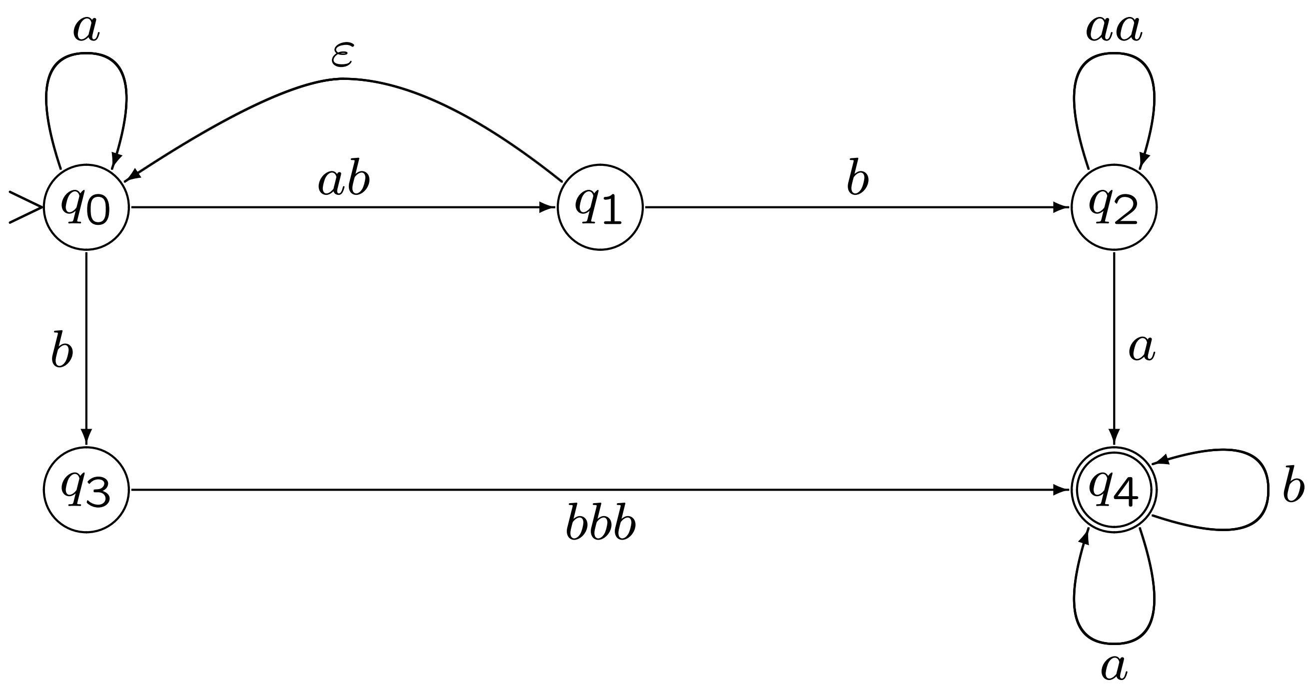
Defining the accepted language

A configuration (q', w') is derivable in one step from the configuration (q, w) by the machine M ($(q, w) \vdash_M (q', w')$) if

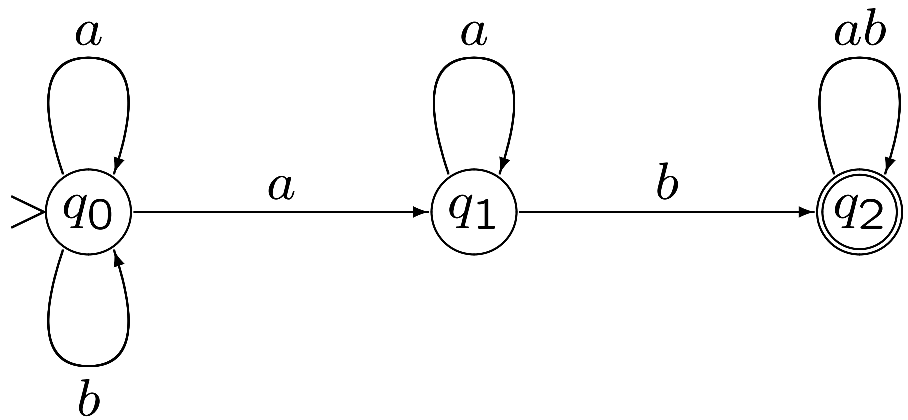
- $w = uw'$ (word w begins with a prefix $u \in \Sigma^*$),
- $(q, u, q') \in \Delta$ (the three-tuple (q, u, q') is in the transition relation Δ).

A word is accepted if *there exists* an execution (sequence of derivable configurations) that leads to an accepting state.

Examples



$$L(M) = ((a \cup ab)^* bbbb\Sigma^*) \cup ((a \cup ab)^* abb(aa)^* a\Sigma^*)$$



$$L(M) = \Sigma^* ab(ab)^*$$

Words ending with at least one repetition of ab.

2.6 Eliminating non determinism

Definition
Two automata M_1 and M_2 are equivalent if they accept the same language, i.e. if $L(M_1) = L(M_2)$.

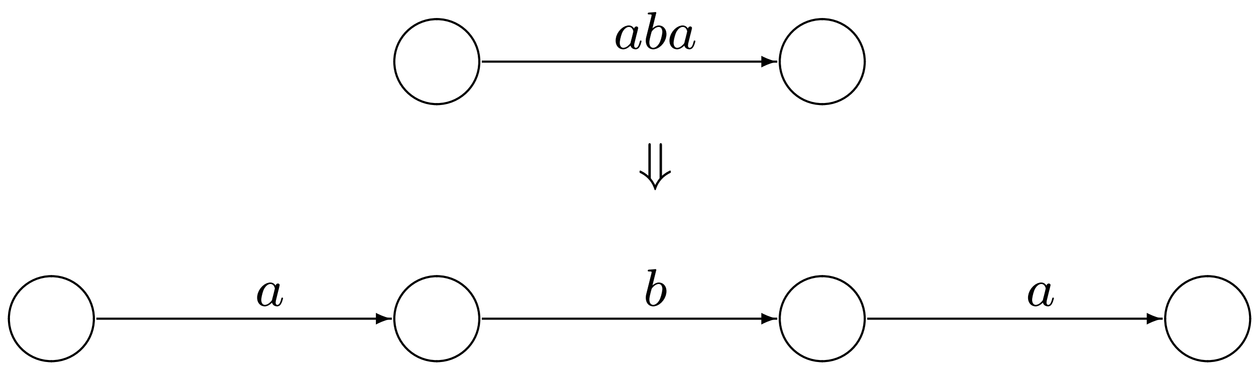
Theorem
Given any nondeterministic finite automaton, it is possible to build an equivalent deterministic finite automaton.

2.6 Idea of the construction

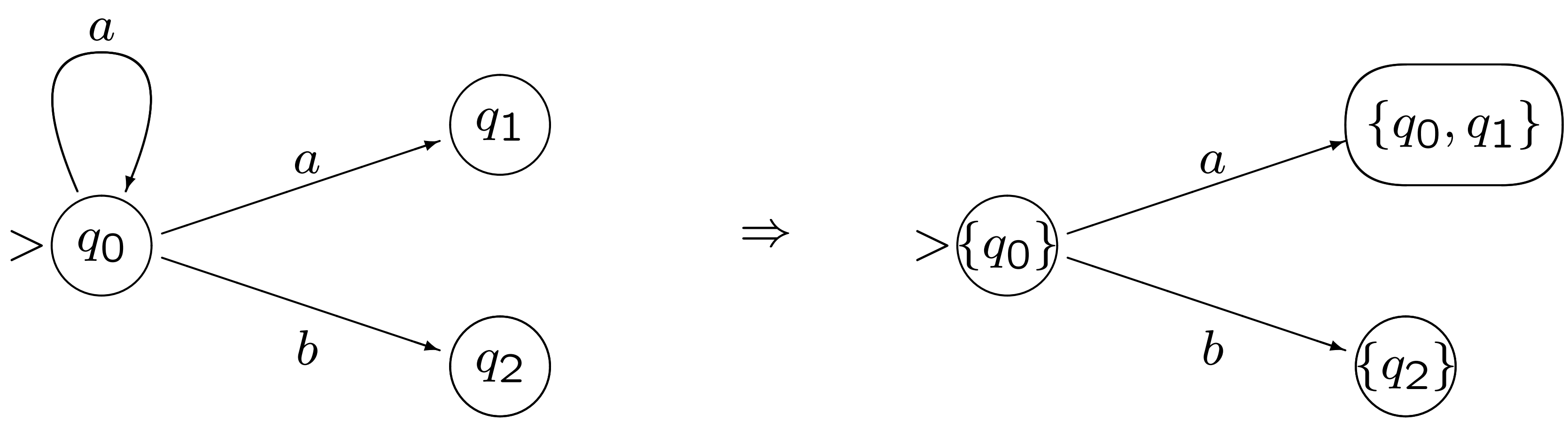
1. Eliminate transitions of length greater than 1.

2. Eliminate compatible transitions

Transitions of length greater than 1



Compatible transitions



Formalization

Non deterministic automaton $M = (Q, \Sigma, \Delta, s, F)$. Build an equivalent non deterministic automaton $M' = (Q', \Sigma, \Delta', s, F)$ such that $\forall (q, u, q') \in \Delta',\ |u| \leq 1$.

- Initially $Q' = Q$ and $\Delta' = \Delta$.
- For each transition $(q, u, q') \in \Delta$ with $u = \sigma_1\sigma_2\ldots\sigma_k$, $(k > 1)$:
  - remove this transition from Δ',
  - add new states $p_1, \ \ldots, \ p_{k-1}$ to Q',
  - add new transitions (q, σ_1, p_1), (p_1, σ_2, p_2), ... , (p_{k-1}, σ_k, q') to Δ'

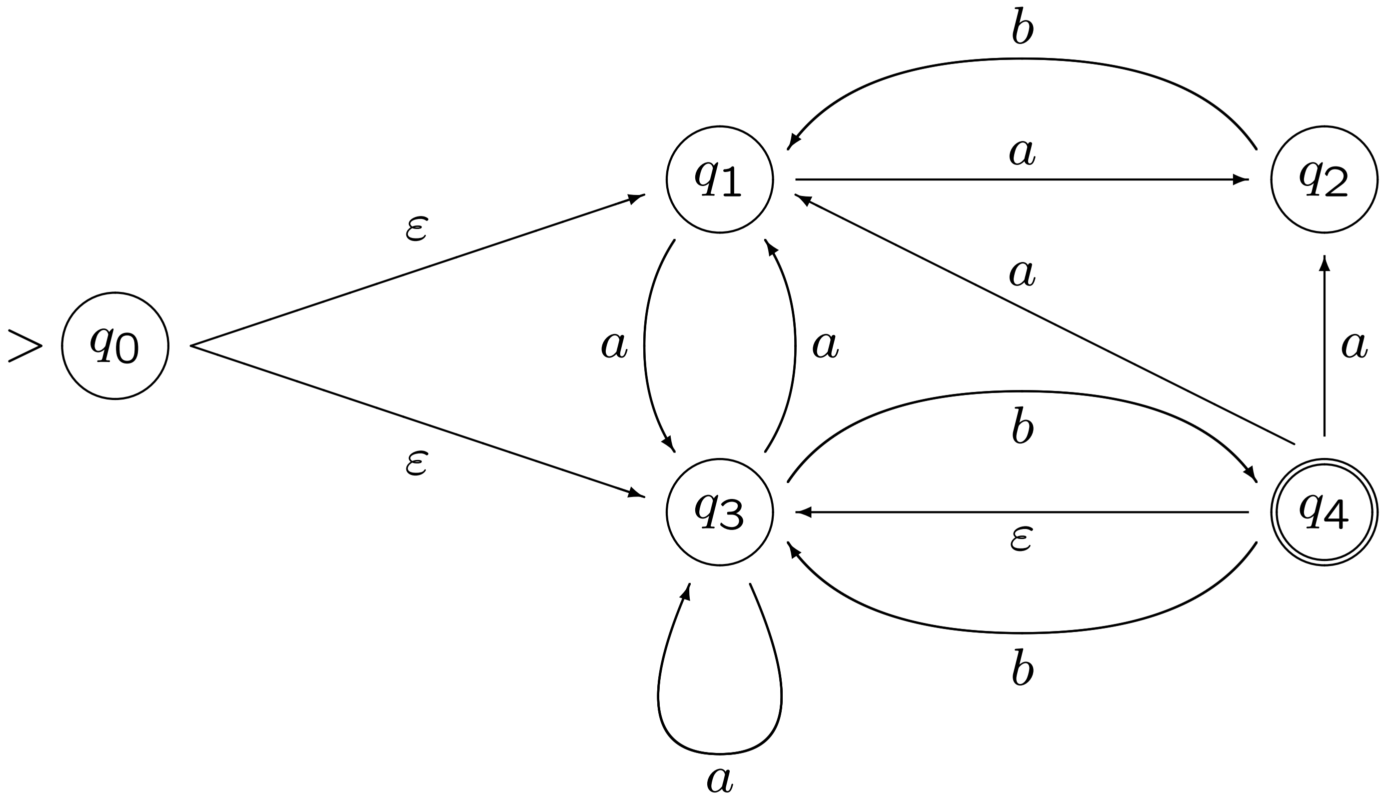
Formalization

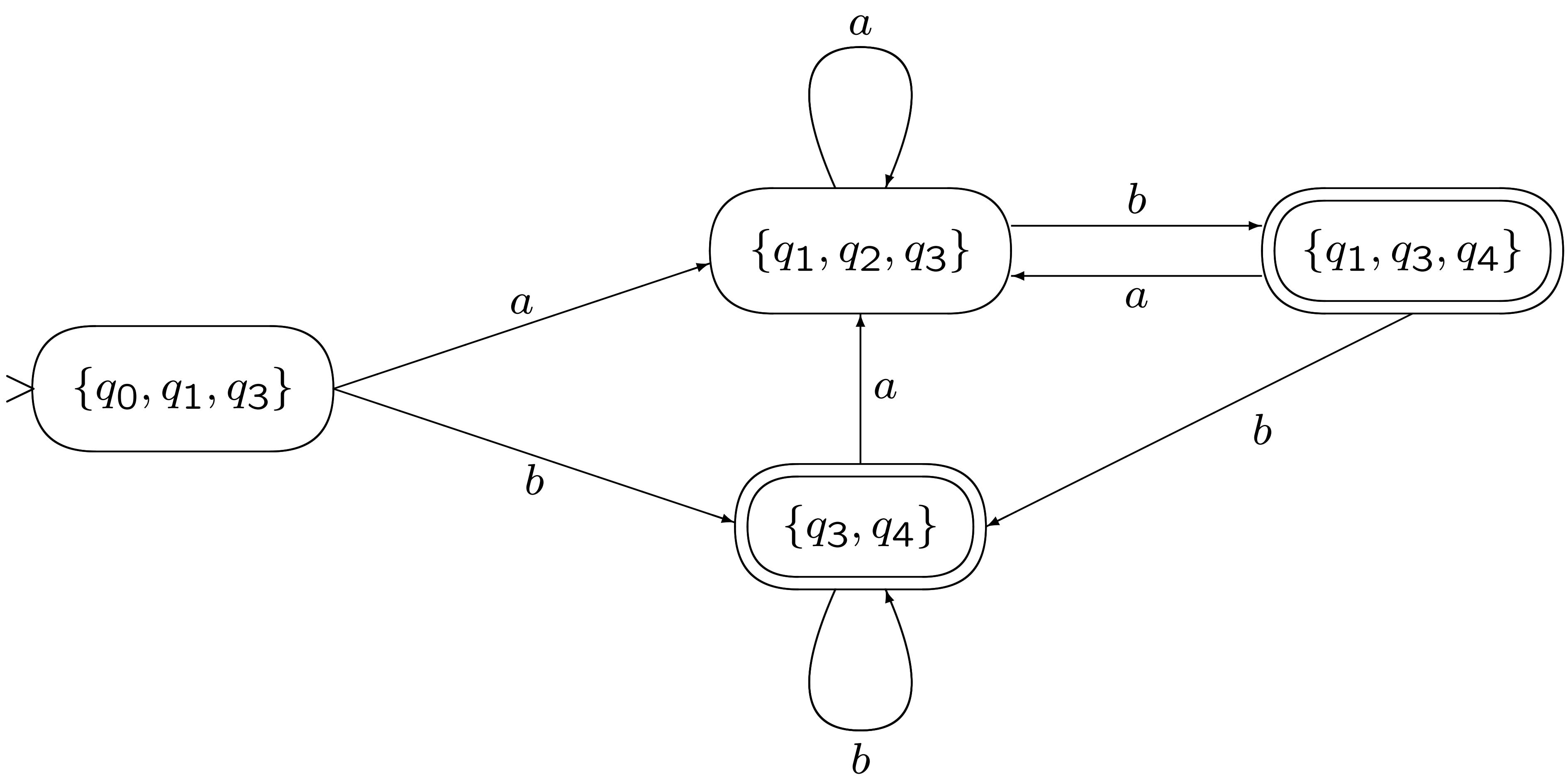
Non deterministic automaton $M = (Q, \Sigma, \Delta, s, F)$ such that $\forall (q, u, q') \in \Delta',\ |u| \leq 1$. Build an equivalent deterministic automaton $M' = (Q', \Sigma, \delta', s', F')$.

$$E(q) = \{p \in Q \mid (q, w) \vdash_M^* (p, w)\}.$$

- $Q' = 2^Q$.
- $s' = E(s)$.
- $\delta'(\mathbf{q}, a) = \bigcup\{E(p) \mid \exists q \in \mathbf{q}\ : (q, a, p) \in \Delta\}$.
- $F' = \{\mathbf{q} \in Q' \mid \mathbf{q} \cap F \neq \emptyset\}$.

Example




a
b
a
a
a
b
b
b
$\{q_1, q_2, q_3\}$
$\{q_1, q_3, q_4\}$
$\{q_0, q_1, q_3\}$
$\{q_3, q_4\}$

Alternative construction: Only accessible states

1. Initially Q' contains the initial state s'.

2. The following operations are then repeated until the set of states Q' is no longer modified.

   (a) Choose a state $\mathbf{q} \in Q'$ to which the operation has not yet been applied.

   (b) For each letter $a \in \Sigma$ compute the state $\mathbf{p}$ such that $\mathbf{p} = \delta'(\mathbf{q}, a)$. The state $\mathbf{p}$ is added to Q'.

2.7 Finite automata and regular expressions

Theorem
A language is regular if and only if it is accepted by a finite automaton.

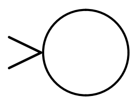
We will prove:

1. If a language can be represented by a regular expression, it is accepted by a non deterministic finite automaton.

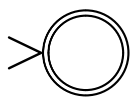
2. If a language is accepted by a non deterministic finite automaton, it is regular.

From expressions to automata

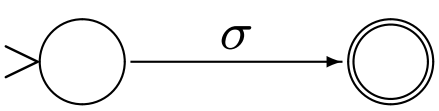
- $\emptyset$



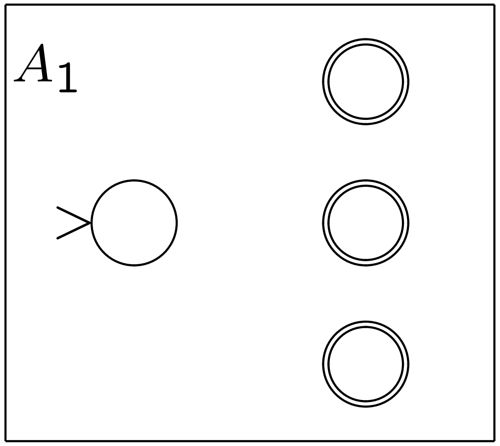
- ε

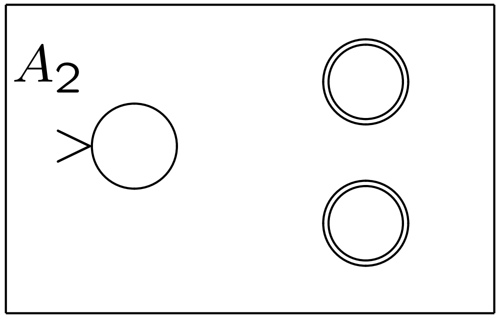


- $\sigma \in \Sigma$

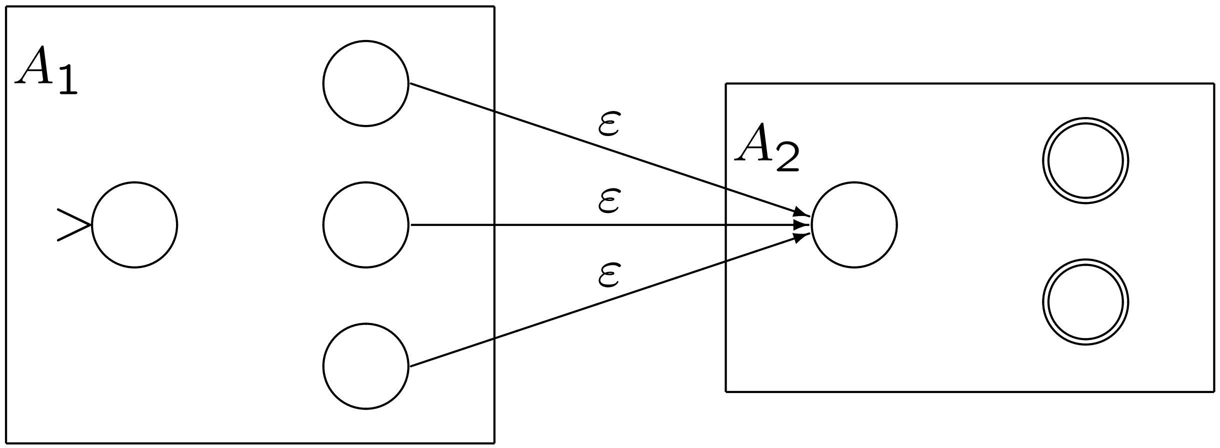


$$\alpha_1 : A_1 = (Q_1, \Sigma, \Delta_1, s_1, F_1)$$
$$\alpha_2 : A_2 = (Q_2, \Sigma, \Delta_2, s_2, F_2)$$





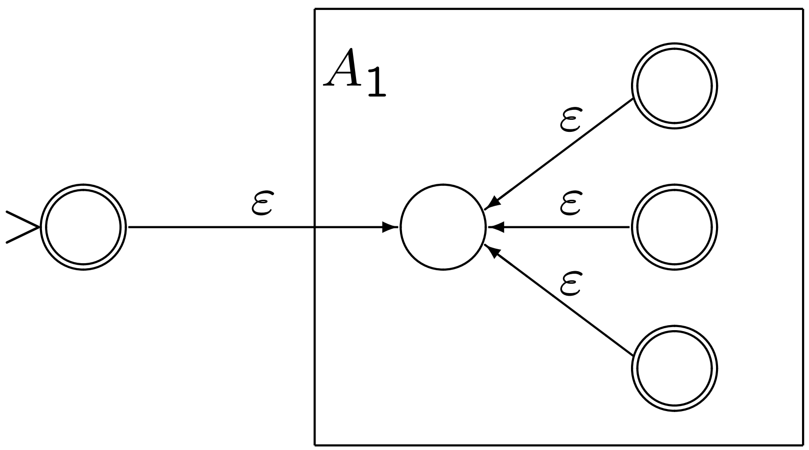
- $\alpha_1 \cdot \alpha_2$



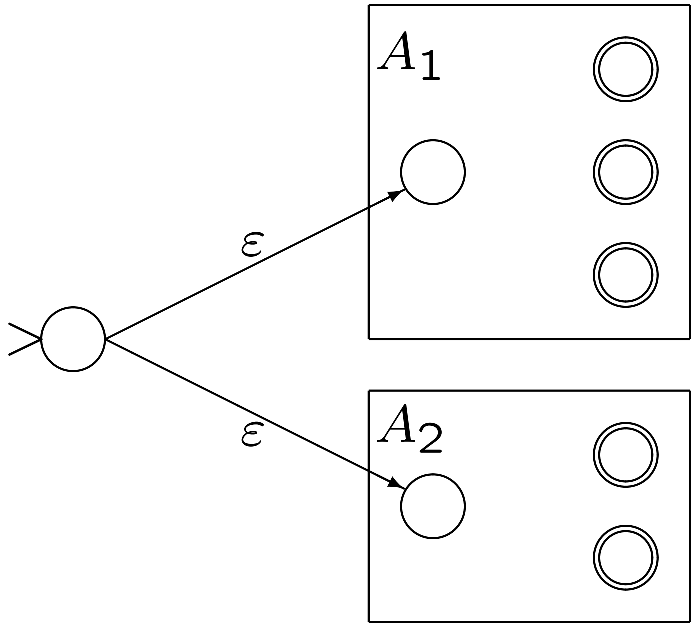
Formally, $A = (Q, \Sigma, \Delta, s, F)$ where

- $Q = Q_1 \cup Q_2$,
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q, \varepsilon, s_2) \mid q \in F_1\}$,
- $s = s_1$,
- $F = F_2$.

- $\alpha = \alpha_1^*$



- $\alpha = \alpha_1 \cup \alpha_2$



From automata to regular languages

Intuitive idea:

- Build a regular expression for each path from the initial state to an accepting state.

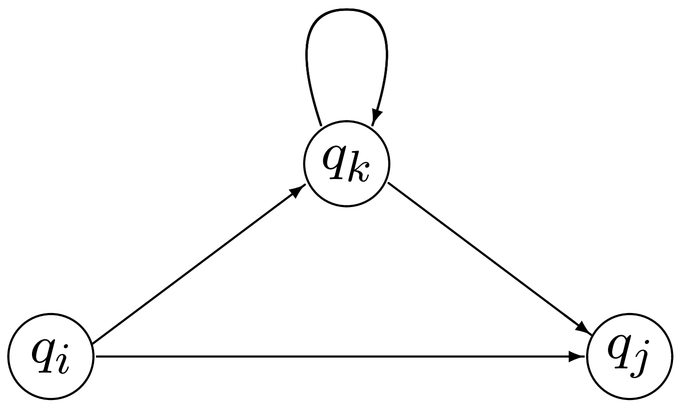
- Use the $*$ operator to handle loops.

Definition

Let M be an automaton and $Q = \{q_1, q_2, \ldots, q_n\}$ its set of states. We will denote by $R(i, j, k)$ the set of words that can lead from the state q_i to the state q_j, going only through states in $\{q_1, \ldots, q_{k-1}\}$.

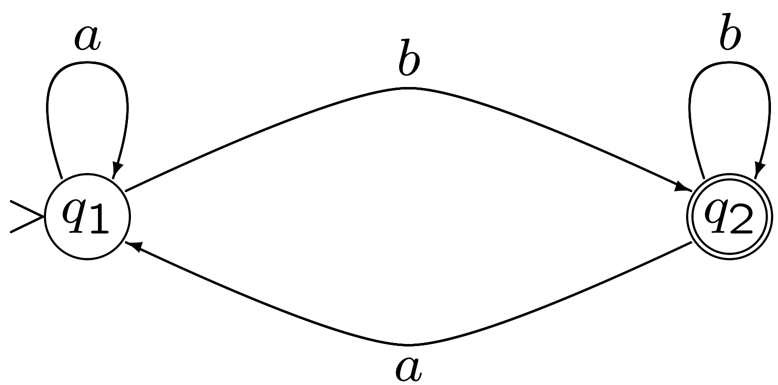
$$R(i,j,1) = \begin{cases} \{w \mid (q_i, w, q_j) \in \Delta\} & \text{si } i \neq j \\ \{\varepsilon\} \cup \{w \mid (q_i, w, q_j) \in \Delta\} & \text{si } i = j \end{cases}$$

$$\begin{aligned} R(i,j,k+1) \;\; = \;\; & R(i,j,k) \cup \\ & R(i,k,k)R(k,k,k)^*R(k,j,k) \end{aligned}$$



$$L(M) = \bigcup_{q_j \in F} R(1, j, n+1).$$

Example



	$k=1$	$k=2$
$R(1,1,k)$	$\varepsilon \cup a$	$(\varepsilon \cup a) \cup (\varepsilon \cup a)(\varepsilon \cup a)^*(\varepsilon \cup a)$
$R(1,2,k)$	b	$b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b$
$R(2,1,k)$	a	$a \cup a(\varepsilon \cup a)^*(\varepsilon \cup a)$
$R(2,2,k)$	$\varepsilon \cup b$	$(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b$

The language accepted by the automaton is $R(1,2,3)$, which is

$$
\begin{aligned}
[b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b] \;\cup\; & [b \cup (\varepsilon \cup a)(\varepsilon \cup a)^* b] \\
& [(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b]^* \\
& [(\varepsilon \cup b) \cup a(\varepsilon \cup a)^* b]
\end{aligned}
$$